AMENDMENTS TO THE SPECIFICATION

Please substitute the following replacement paragraphs for like-numbered paragraphs of the specification:

[0010] Each data value stored in the memory 133 is referred to herein as an entry (shown generally at 132) and includes at least a key, KEY, and validity value, V. The key is formed from selected bits within the input data value and is input, at least in part, to the hash index generator 131 to generate the hash index at which the entry is stored (i.e., the storage index) . The validity value may include one or more bits and is used to indicate the presence of a valid entry. An entry 132 may optionally include mask information, MSK (e.g., a mask value to be applied during a comparison operation, or an encoded value that can be decoded to generate a mask value), a priority value, PRI, which is a numeric value that indicates a priority of the entry relative to other entries, and control/configuration information, CNTRL/CFG, that may be used to control certain operations within the hash CAM device 130. The priority values included within the various entries within the memory 133 may be assigned in ascending priority order (i.e., lower numeric values indicate a higher priority than higher numeric values) or descending priority order (i.e., higher numeric values indicate a higher priority than lower numeric values). In the description that follows ascending priority order is generally assumed, though descending priority order may be alternatively be used. Also, in alternative embodiments, non-numeric values may be used to indicate priority (e.g., codewords or other indications of priority may be used).

[0040] Note that the compare cells 257_1-257_S may alternatively pull the match line low in response to detecting that the search key component is less than the entry key component (i.e., signal a less-than (LT) condition). Referring again to Figure 13, for example, instead of comparing the search key bit against the complement of the stored key in transistors 262 and 264, the complement search key bit may be compared against the uncomplemented entry key bit, pulling line 260_i (now a LT line) low if the entry key bit is high and the search key bit is low. Similarly transistors 268 and 266 may pass a LT_{i-1} signal from a less significant compare cell to the LT line 260_i if the complement search key bit is high (meaning that the search key bit is low) or the entry key bit is high.

[0043] Reflecting on the operation of the hash CAM devices of Figures 3 and 6 it should be noted that, in an alternative embodiment, the information maintained in the configuration register 137 may be stored on an entry-by-entry basis within memory 133, effectively enabling multiple search tables to be stored within a single device. In such an embodiment, a hash index may be generated based on a predetermined portion (or all) of a search value. The configuration information within the entry retrieved from the indexed location may then be used to qualify the comparison of the search value and the entry. For example, an entry type within the entry may be compared with a class code associated with the search value to qualify any match detection (i.e., suppress assertion of the match flag 136 if the entry type and class code do not match), mask information within the entry may be used to mask mismatches at selected bit positions of the entry and search key, control information within the entry may be used to select the type of comparison operation to be performed (e.g., binary, ternary and/or range compare) and so forth. In general, any information that may be stored in the configuration register may be stored on an entry by entry basis within the memory 133.

[0063] Figure 19 illustrates an embodiment of the device priority logic 325 of Figure 15 (other embodiments may be used). (Note that the device priority logic 325 may also be used in the hash CAM block of Figure 14; a case in which there is only one hash CAM block). The device priority logic 325 includes a priority compare circuit 361, priority encoder 363 and index selector 365. As discussed in reference to Figure 15, the device priority logic 325 receives the block flags, block priority values and block indices from each of N hash CAM blocks (i.e., BF_1-BF_N, BP_1-BP_N, and BIN_1-BIN_N) and also the block flag, block priority value and block index from the overflow CAM device (BF_{OFC},BP_{OFC},BIN_{OFC}), and outputs a highest priority one of the block priority values, for which the corresponding block flag signal is asserted, as a device priority number 306 (DP). Herein, a block priority value for which the corresponding block flag signal is asserted is referred to as a match-qualified block priority value, and, conversely, a block priority value for which the corresponding block flag signal is deasserted is referred to as a disqualified block priority value. During a given NFA or search operation, disqualified block priority values are disabled from participation in a priority number comparison operation (e.g., by being forced to a lowest possible priority value) such that only a match-qualified block priority value may be output as the device priority number. In one embodiment, if all the block priority values are disqualified (i.e., no match signals are asserted), a lowest possible priority number is output as the device priority number 306.

[0073] Because all four hash CAM blocks 319_1-319_4 are initially empty, each hash CAM block will respond during a NFA operation by asserting its block flag and outputting the same fixed block priority value. Thus, the device priority logic (i.e., element 325 of Figures 15 and 19) will select one of the hash CAM blocks according to a predetermined, hardwired tie resolution policy. In this particular example, lower numbered hash CAM blocks (e.g., those with lower physical addresses) are given higher priority over higher numbered hash CAM blocks so that, even though all four hash CAM blocks output the same priority number, the device priority logic will generate a block identifier (BID) that corresponds to hash CAM block 1 (319_1). (In alternative embodiments, different tie resolution policies may be used, and a value within the block configuration register may be used to select between multiple tie resolution policies). Thus, in the example of Figure ~~24~~25, hash index 340 and a block identifier for hash CAM block 1 form the device index output by the CAM device 429. Accordingly, during an ensuing insert operation, an entry is stored in hash CAM block 1 at the location pointed to by hash index 340. This location within the hash CAM device (i.e., location 430) is referred to herein as an insertion address, and is indicated in Figure ~~24~~ 25 by a lined shading pattern~~-~~.

[0078] Figure 29 illustrates an alternative embodiment of a CAM device 450 that includes multiple hash CAM blocks 445. In the CAM device 450, the hash CAM blocks 445 are organized into G groups of M hash CAM blocks each, with each hash CAM block operating as described in reference to the previous figures, and with each group of hash CAM blocks sharing a respective assembler/mask circuit 441_1-441_G and hash index generator 443_1-443_G. A group configuration register 447_1-447_G is associated with each group of hash CAM blocks and used to control the operation of the corresponding assembler/mask circuit. Figure 30 illustrates a more detailed embodiment of the assembler/mask circuit 441 and hash index generator 443 of Figure 29. As shown, the assembler/mask circuit 441 includes K assembler/mask subcircuits, 451_1-451_K, each coupled to provide a respective masked key, 194_1-194_K, to a corresponding one of K hash function circuits 453_1-453_K within the hash function generator 443. Each assembler/mask subcircuit includes an assembler subcircuit and key mask logic subcircuit that operate in the manner described above in reference to the assembler circuit 191 and key mask logic 193 of Figure 6 to generate an assembled, masked key. In one embodiment, the group configuration register 447 includes a set of K entry type values that are input, respectively, to the K assembler/mask subcircuits 451_1-451_K to enable each assembler/mask subcircuit to be configured to output a different masked key based on the same incoming data value. By this arrangement, each hash function circuit 453_1-~~543_K~~ 453_K may receive a different masked key. Each of the M hash CAM blocks within a given group (i.e., 445_1-554_M) includes a configuration circuit that may be programmed with a respective hash function select value (HF_SEL) and includes a hash function select 449_1-449_M circuit to select, according to the programmed hash function select value, one of the K hash indices, HI_1-HI_K, generated by the hash index generator 443. Accordingly, each hash CAM block within the group may receive a hash index which has been generated according to a selected entry type, key mask and hash function. Comparing the hash CAM device 450 to the hash CAM device of Figure 15, the total number of hash index generator circuits is reduced by a factor of M while retaining independent configurability for each hash CAM block 445. Note that in an alternative embodiment, the groups of hash CAM blocks depicted in Figure 29 need not all include the same number of CAM blocks. Also, while embodiments of hash CAM blocks are described below as including a dedicated assembler circuit, key mask logic and hash index generator, such hash CAM blocks may be modified to operate with the shared assembler/mask circuit 441 and shared hash index generator 443 of Figures 29 and 30.

[0079] Figure 31 illustrates an embodiment of a hash CAM block 470 that includes a segmented memory 471 and which may be used within any of the multiple hash CAM block devices or single hash CAM block devices described above. The hash CAM block 470 includes an assembler circuit 191, key mask logic 193, hash index generator 131 and block select logic ~~131~~ 331 that operate generally as described above in reference to Figure 16. The segmented memory 471 includes an address decoder 195, read write circuit 481 and segmented memory array 477. The address decoder 195 responds to a hash index 340 from the hash index generator 131 by activating one of a plurality of word lines 204 in the manner described above in reference to Figure 6. Each word line 204 is shared among each of four segments 479_1-479_4 within the segmented memory array 477 (more or fewer segments may be used to form the memory array in alternative embodiments), so that the activated word line enables read and write access to a row of storage cells that spans all four segments. During a NFA or search operation, the contents within the entire row of storage cells (i.e., the contents of the word-line indicated location within each of segments 479_1-479_4) are sensed by a set of four sense amplifier banks (SA) within the read/write circuit 481 and output as a set of four entry segments 138_1-138_4 to an output logic circuit 475. During an insert operation, a segment address 478 (SA) is provided to the read/write circuit 481 to select one or more of write driver banks (WD) within the read/write circuit 481 to store an entry within the corresponding segment or segments 479.

[0080] The hash CAM block 470 includes a configuration register 473 that is similar to the configuration register 137 described in reference to Figures 6 and 16, except that the entry type value within the configuration register includes an entry size field which specifies the size (i.e., number of constituent bits) of entries stored within the segmented memory 471. In the embodiment of Figure 31 for example, entries which span 1, 2, or 4 segments may be stored within the segmented memory. Such values are referred to herein as x1, x2, and x4 entries and correspond to x1, x2, and x4 operating modes of the CAM block 470 selected by the entry size field. In alternative embodiments that include more or fewer memory segments 479, entries may span different numbers of segments.

[0087] The search priority logic 550 includes logic OR gates 541, 543, 545 and 547 to generate match-qualified priority values; comparator circuits C_1, C_2 and C_3 to compare the match qualified ~~priorities~~priority values; multiplexers 549, 551 and 553 to select a search priority value from among the match qualified priority values; and AND gates 555, 557, 559 and 561 to output the index select signals IS_1-IS_4 via multiplexer bank 570. Each of the OR gates 541, 543, 545, 547 outputs a respective match qualified priority QP_{S1}, QP_{S2}, QP_{S3}, QP_{S4} according to the state of the corresponding segment flags SF_1-SF_4. For example, if segment flag SF_1 is high (indicating a match condition), OR gate 541 will output the PRI_{S1} value as the match-qualified priority value QP_{S1}. If SF_1 is low, indicating a mismatch condition, OR gate 541 will output a match disqualified priority value QP_{S1} in which all constituent bits are high (i.e., due to the inversion of the segment flag at the input to OR gate 541), which, in the embodiment of Figure 34, is the lowest possible priority value. OR gates 543, 545 and 547 are additionally coupled to receive the x4 signal so that, in the x4 mode, the priority values QP_{S2}, QP_{S3} and QP_{S4} are driven to the lowest possible priority value (i.e., disqualified because they are unused). Consequently, in the x4 configuration, the match qualified priority values for segments 2-4 of the segmented memory will not exceed the match qualified priority value for segment 1 and therefore are effectively ignored. That is, the segment 1 priority value, PRI_{S1}, is the priority value for a x4 entry. In alternative embodiments, any of others of the segments, including combinations of two or more segments may be used to source the x4 priority value. OR gates 543 and 547 are further coupled to receive the x2 signal so that, in the x2 mode, the priority values for segments 2 and 4 are ignored, the x2 entries being represented by priority values stored in segments 1 and 3. In alternative embodiments, others of the segments, including combinations of segments may be used to source the x2 priority values and/or the x4 priority value.

[0090] Figure 35 illustrates alternative NFA priority logic 580 that may be used in place of NFA priority logic 571 of Figure 34. The NFA priority logic 580 generates an insert priority value 582 having an R-bit fill count component 586 and at least one bit 588, referred to herein as a partial fill bit, that indicates whether or not at least one segment of the indexed memory row is occupied by at least one valid entry (i.e., a partially filled row). The R-bit fill count 586 is generated by a fill counter 585 which is incremented in response to an insert operation and decremented in response to a delete operation (i.e., an operation to invalidate an entry). Accordingly, the fill count 586 indicates the number of valid entries stored in the memory of a hash CAM block such that, in an embodiment in which numeric value is inversely proportional to priority, the less filled of two CAM blocks having matching partial fill bits 588 will output a higher priority value than the more filled CAM block. In the embodiment of Figure 35, insert and delete signals (INS and DEL) are applied, respectively, to up and down inputs of the fill counter 585, the insert and delete signals being generated, for example, by an instruction decoder. In an alternate embodiment, the fill counter 585 may count down in response to the insert signal and up in response to the delete signal so that the less filled of two CAM blocks having matching partial fill bits 588 will output a numerically higher (lower priority) priority value than the more filled CAM block.

[0092] Figure 36 illustrates an embodiment of the segment index encoder 493 of Figure 32. The segment index encoder 493 receives the index select signals, IS_1-IS_4, from the segment priority logic and selects one of four predetermined segment indices, 00b, 01b, 10b or 11b (the 'b' indicating binary notation) to be output as the segment index 488. Each of the predetermined indices is an address to one of the four different segments within the segmented memory. Thus, if during either a NFA operation or a search operation, index select signal IS_1 is asserted, value 00 at port P1 of a multiplexer 581 is output as the segment index 488. That is, if segment 1 within the segmented memory is signaled to be unoccupied in a NFA operation, or signaled to contain a matching entry in a search operation, the address of that segment (i.e., 00b) will be output as the segment index 488. In one embodiment, the multiplexer 581 is designed to resolve ties (i.e., more than one index select signal being asserted) in favor of the lowest numbered of the index select signals. For example, if all four index select signals are asserted, the segment index that corresponds to index select signal IS_1 (i.e., 00b) is output as the segment index 488. Different tie resolution schemes may be used in alternative embodiments.

[0096] Figures 39 and 40 illustrate NFA operations within a hash CAM device 594 that is identical to the exemplary hash CAM device 592 of Figures 37 and 38 except that the hash CAM device 594 includes the capacity-dependent NFA priority logic 580 of Figure 35. Referring specifically to Figure 39, hash CAM blocks 1, 2, 3 and 4 are all indexed at completely unfilled rows so that that the partial fill bit of the insert priority value for each hash CAM block is set. Because the fill level of hash CAM block 2 is lower than that of hash CAM blocks 1, 3 and 4, however (hash CAM block ~~1~~2 contains only one entry, while hash CAM blocks ~~2~~1, 3 and 4 each contain more than one entry), hash CAM block 2 outputs a numerically lower (higher priority) priority value than hash CAM blocks 1, 3 and 4 and is therefore selected for entry insertion.

[0099] Figure 41 illustrates another embodiment of a hash CAM block 600 which may be used within any of the multiple hash block CAM devices or single hash CAM block devices described above. The hash CAM block 600 includes an assembler circuit 191, key mask logic 193, hash index generator 131, block select logic 331 and configuration register 473, all of which operate as generally described above in reference to Figure 31. The hash CAM block 600 additionally includes a binary CAM circuit 601 to perform a function referred to herein as indirect hashing. In indirect hashing, ~~in the hash CAM block 600,~~ the hash index 340 generated by the hash index generator 131 is not used to index the memory 477 directly, but rather is compared with stored hash indices within the binary CAM 601 to identify a matching hash index within a storage row of the binary CAM 601, and to select a corresponding row within a memory array 477.

[0100] Figure 42 illustrates the different results produced by direct hashing, in which a hash index is used to address the memory array 477 directly, and indirect hashing achieved through use of the binary CAM 601. As shown, the hash index generator 131 generates a hash index 340 having N constituent bits. Because the memory array 477 only has 2^M storage rows, however, only M of the N bits within hash index 340 may be used to directly address the memory array 477. That this, in a direct hashing scheme, either the hash index generator 131 generates a hash index having a number of bits that corresponds to the size of the memory to be addressed, or the hash index is truncated to the appropriate number of bits before addressing the memory array. The truncation of T bits from the N-bit hash index is shown in the direct hashing example of Figure 42 (i.e., N=M+T). The truncation of T bits reduces the overall hashing space by a factor of 2^T, thereby increasing the likelihood of collisions relative to indirect hashing. By contrast, in the indirect hashing approach, the full set of N bits may be used (or at least more than M bits) to select a row of the memory array 477, even though the memory array 477 has fewer than 2^N storage rows. As shown, the binary CAM 601 is used to store the full N-bit hash index within a selected row of CAM cells during an insert operation, the selected row of CAM cells corresponding to a row within the memory array in which the entry is inserted. In a search operation, the hash index generated by the hash index generator 131 is compared with each of the hash indices stored within the binary CAM 601 to determine whether the hash index has previously been loaded in to the binary CAM 601. If a match is found, a match line corresponding to the row of the binary CAM 601 containing the matching hash index is activated. In one embodiment, the match line is used to drive a corresponding word line within the memory array. By this arrangement, a matching hash index within the binary CAM enables read or write access to the corresponding storage row within the memory array 477. Thus, the binary CAM effectively presents a larger hash index space to the address generation circuitry (i.e., the hash index generator 131) than is provided by the underlying memory array 477. Because the full hash index 340 may be used (that is, rather than a truncated hash index), the likelihood of collisions between incoming data values is substantially reduced (i.e., as compared with direct hashing). Further, as shown in Figure 42 the binary CAM may be loaded in a predetermined manner (e.g., bottom to top, top to bottom, etc.) such that the full space within the memory array may be used. This is in contrast to the direct hashing approach~~,~~ in which the memory array may be filled only up to a certain percentage before collisions become so statistically likely that the hash CAM block is, for practical purposes, full.

[0102] In applications in which insertion policy favors insertion at partially filled rows, it may be desirable to increase the likelihood of a match within the binary CAM array 625 by masking selected bits of the hash index, effectively selecting a smaller hash index. In the embodiment of Figure 43, a hash index mask circuit 641 is provided to mask selected bits of the hash index 340 in accordance with the entry size value within the block configuration register. As a specific example, for a x1 configuration, a truncated, 12-bit hash index 340 is output from the hash index generator 131 and four of the bits of the hash index 340 are masked by the hash index mask circuit 641 hash for x1 entries; two bits of the hash index 340 are masked by the hash index mask circuit 641 for x2 entries, and none of the bits of the hash index 340 are masked by the hash index mask circuit 641 for x4 entries. This hash index masking policy reflects the significance of partially filled rows for the x1, x2 and x4 configurations. That is, there can be no partially filled row in the x4 configuration (row is either entirely occupied or unoccupied), and partially filled rows are statistically more likely in the x1 configuration than the x2 configuration due to the fact that a partially filled row remains partially filled in the x1 configuration until it has been selected for entry insertion four times (versus two times for the x2 configuration). Different masking levels may be used in alternative embodiments, including embodiments having more or fewer entry segments and therefore different configurations in which partially filled rows are possible. Also, different hash index truncation selections may be enabled in different hash CAM blocks within a given device. For example, in one embodiment some of the hash CAM blocks include memory arrays (and binary CAM arrays) having 512 storage rows, while others of the hash CAM blocks include memory arrays (and binary CAM arrays) having 256 storage rows. Different hash index truncation selections may be programmed for the different hash CAM blocks according to the total addressable storage space (and therefore the storage capacity) of the memory array.

[0120] If a match is not detected within the binary CAM (717), then at 721 a buffered binary CAM flag (stored during the most recent binary CAM load operation) is inspected to determine whether the binary CAM is full. If the binary CAM is full, the block flag is deasserted at 727. If the binary CAM is not full, then at ~~735~~ 725 the binary CAM fill count and a logic high partial fill bit (indicated by the summation of 200 hex and the fill count in the example of Figure 49) are output as the block priority value, the block flag is asserted, and the next free binary CAM address (NFBA) and predetermined segment index (00 in this example) are output as the block index.

[0129] Each of the Y tag logic circuits 847 receives a respective match signal 810 from a corresponding row of the CAM array. In one embodiment, each of the match signals 810 includes five component match signals MT and M1-M4, that correspond to the tag segment and entry segments S1-S4, respectively. Each of the tag ~~match~~ logic circuits 847_1-847_Y outputs a respective set of four tag match signals, TM1-TM4, with each of the tag match signals being asserted if (1) a tag value stored in the tag segment matched a class code value associated with the search key (i.e., the class code value (CC) discussed above in reference to Figure 17), and (2) an entry stored in the corresponding entry segment (or, in the x2 and x4 configurations, in a group of two or four entry segments) matched the search key. The entry size information, CFG 840, is input to the tag logic circuits 847_1-847_Y to qualify the detection of a match condition.

[0140] Figure 56 shows one embodiment of a circuit 930, referred to herein as a priority logic element or priority cell, for implementing the truth table of Table 2. The priority cell 930 includes compare circuit 933, isolation circuit 931, and memory element ~~872~~871. Compare circuit 933 is one embodiment of compare circuit 906, and isolation circuit 931 is one embodiment of isolation circuit 904. The priority cell 930 may be used to implement all the priority cells in the priority index table.

[0148] Referring again to Figure 53, the column priority logic 843 compares the column priority values received from the priority array 841 to generate a search priority value, SP, that is the highest priority one of the column priority values. The column priority logic 843 further generates a set of four segment enable signals 852 SE[4:1], each segment enable signal 852 being asserted or deasserted according to whether a corresponding one of the four column priority storage circuits contains a priority value equal to the search priority value. Thus, in the embodiment of Figure 53, if only one column of priority storage circuits contains a priority value equal to the search priority value, then only one of the four segment enable signals 852 will be asserted. Conversely, if more than one column of priority storage circuits contains a priority value equal to the search priority value, then more than one of the four segment enable signals 852 may be asserted. The row logic circuits 845_1-845_Y receive respective sets of the prioritized match signals 844_1-844_Y from the priority array 841 and output, according to the segment enable signals 852, respective sets of qualified match signals 812_1-812_Y, each set of qualified match signals including four component match signals, QM1-QM4. In one embodiment, the column priority logic 843 is implemented in the same manner as the search priority logic 550 of Figure 34, except that the segment flag signals and OR gates 541, 543, 545 and 547 are omitted (i.e., the column priority values are input directly to the comparators); the outputs of AND gates 555, 557, 559 and 561 corresponding to the segment enable signals 852 SE[4:1]~~SE1-SE4~~. In alternative embodiments, other circuits may be used to implement the column priority logic 843 including, without limitation, a priority array as described in reference to Figures 52-54 in which the priority values are supplied from priority array 843 rather than being self contained.

[0151] If the entry size information 840 indicates a x2 configuration, the output of AND gate 953 is selected to drive tag match signals TM1 and TM2, and the output of AND gate 954 is selected to drive tag match signals TM3 and TM4. AND gate 953 performs a logical AND of the outputs of AND gates 951_1 and 951_2 and therefore outputs a logic high signal if (1) the stored tag matches the class code associated with the search key, and (2) a x2 entry stored in segments 1 and 2 matches the search key (as indicated by assertion of signals M1 and M2). If the stored tag does not match the class code or if either of entry segment match signals M1 and M2 indicate a mismatch condition, tag match signals TM1 and TM2 are both deasserted to indicate a mismatch. AND gate 954 performs a logic AND of the outputs of AND gates 951_3 and 951_4 and therefore outputs a logic high signal if (1) the stored tag matches the class code associated with the search key, and (2) a x2 entry stored in segments 3 and 4 matches the search key (as indicated by assertion of signals M3 and M4). If the stored tag does not match the class code or if either of entry segment match signals ~~M1~~ M3 and ~~M2~~ M4 indicate a mismatch condition, tag match signals ~~TM1~~ TM3 and ~~TM2~~ TM4 are both deasserted to indicate a mismatch.

[0153] Figure 60 illustrates an embodiment of the row logic 845 of Figure 53. As discussed above, the column priority logic 843 compares the column priority values (CP1-CP4) received from the priority array to generate a plurality of segment enable signals 852_1-852_4 (SE1-~~SEZ~~SE4), each segment enable signal 852 indicating whether the corresponding column priority value is equal to the search priority value 832 (SP) (i.e., a highest priority one of the column priority values). The row logic 845 includes AND gates 956_1-956_4 to logically AND the prioritized match signals PM1-PM4 output from a given row of the priority array with the segment enable signals SE1-SE4. The output of each logic AND gate 956 is coupled to a first input port of a respective one of multiplexers 957_1-957_4, so that, when an operation select signal (OPSEL) indicates a search operation or NFA operation, the outputs of AND gates 956_1-956_4 are selected to be the qualified match signals 812 (QM1-QM4). Thus, during a search or NFA operation, qualified match signal QM1 will be asserted if the corresponding prioritized match signal (PM1) is asserted and segment enable signal SE1 is asserted. That is, qualified match signal QM1 will be asserted if PM1 indicates that the corresponding priority value is equal to the column priority value for column 1 of the priority array, and SE1 indicates that the priority value is equal to the search priority 832 (i.e., the priority value is an intra-column and inter-column winner of priority comparisons). Qualified match signals QM2, QM3 and QM4 are similarly asserted if their corresponding priority values are intra-column and inter-column winners of priority comparisons.

[0154] Still referring to Figure 60, if the operation select signal (OPSEL) indicates a write to the CAM array, then validity values V1-V4 are selected by multiplexers 957_1-957_4 to be output as qualified match signals QM1-QM4 instead of the signals generated by AND gates 956. In one implementation, the validity values are active low signals which, if high, indicate that the corresponding entry segment within the CAM array does not have a valid entry stored therein. That is, the validity values may be interpreted as active high unoccupied (empty) signals. Accordingly, when the validity values are selected to be output as the qualified match signals, QM1-~~QMZ~~QM4, the qualified match signals effectively represent a set of empty flags for the CAM array. The empty flags may be used within the priority encoder 805 of Figure 52 to generate an index of a next free address within the CAM array 801. In one embodiment, the validity values are formed by one or more bits stored in the CAM array 801 within the corresponding entry segment. In an alternative embodiment, to facilitate circuit layout, validity storage circuits are provided both in the CAM array and in a location physically near the multiplexers 957. In the latter embodiment, the validity values stored in the validity storage circuits located near the multiplexers 957 mirror the values stored within the CAM array and are used to drive the qualified match signals when a write operation is selected. In an embodiment in which a predetermined entry value (e.g., code) is used to indicate the presence or absence of a valid entry within an entry segment, the multiplexers 957 may be omitted altogether.

[0168] Numerous functions described as being implemented within the CAM devices of Figures 3-64 may alternatively be implemented by the host processor 975 in response to instructions (i.e., software or firmware) stored within the program store 979. For example, in one embodiment the fill level of the binary CAM (or ternary CAM) within each hash CAM block is tracked by host processor 975 under program control. Each time a hash CAM block is selected for entry insertion at an unfilled row (an event that involves loading a new hash index into the binary CAM), the binary CAM fill count value for the hash CAM block is incremented. Similarly each time a binary CAM entry is deleted (e.g., due to deletion of an entry or entries in a corresponding row of the memory array), the binary CAM fill count value for the hash CAM block is decremented. Prior to selecting a hash CAM block for entry insertion at an unfilled row (a circumstance indicated by a device priority value having, for example, a logic high partial fill bit), the host processor inspects the binary CAM fill count value for the hash CAM block to determine if the count value has reached a maximum (i.e., completely full) level. If the count value indicates a full binary CAM, the host processor does not store the candidate entry in the indicated hash CAM block. Other functions within the CAM devices of Figures 3-64 may similarly be implemented by the host processor 975 under program control.

AMENDMENTS TO THE DRAWING

Please amend Figure 6 to include reference numeral 136.

Please amend Figure 7 to resize the dashed outline box labeled "Assembler 191."

Please amend Figure 30 to include reference numerals 451_1 and 453_1.

Please amend Figure 36 to include reference numeral 488.

Please amend Figure 63 to include reference numeral 970.

Annotated, Marked-up Drawings illustrating the above amendments in red ink are enclosed. Replacement Sheets including all amended Figures are also enclosed.